DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CHANDIGARH UNIVERSITY Discover. Learn. Empower.

Assignment -6

Student Name: Anirudh Gautam **UID: 22BCS12994**

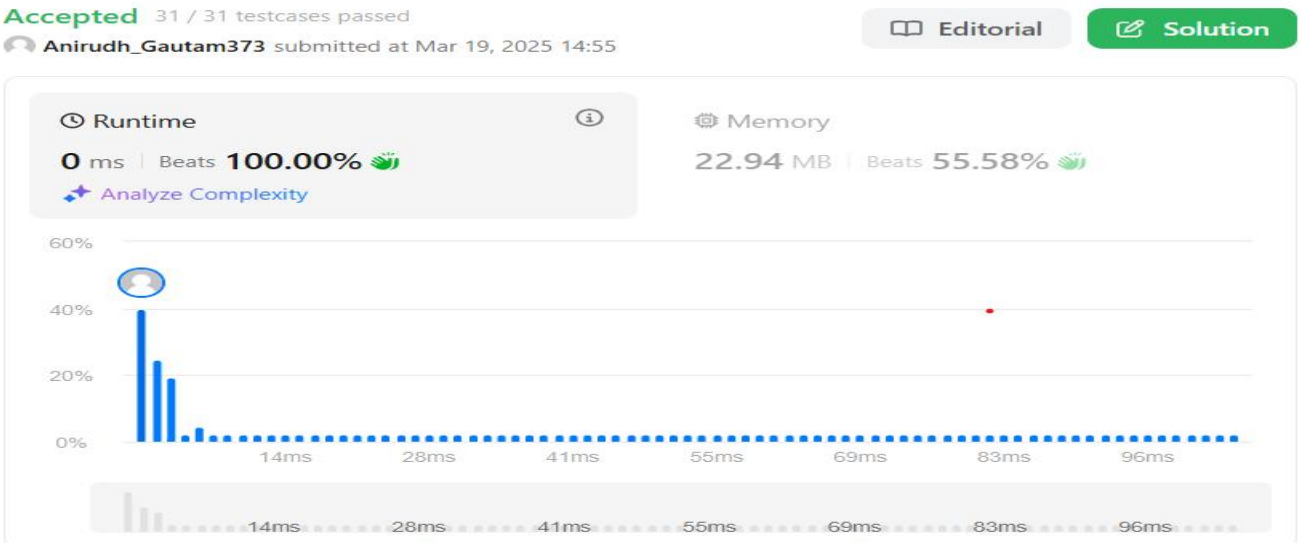
Branch: CSE **Section/Group:FL-602-A**

Semester: 6 **Date of Performance:18/03/25**

Subject Name: Advanced Programming **Subject Code: 22CSH-359**

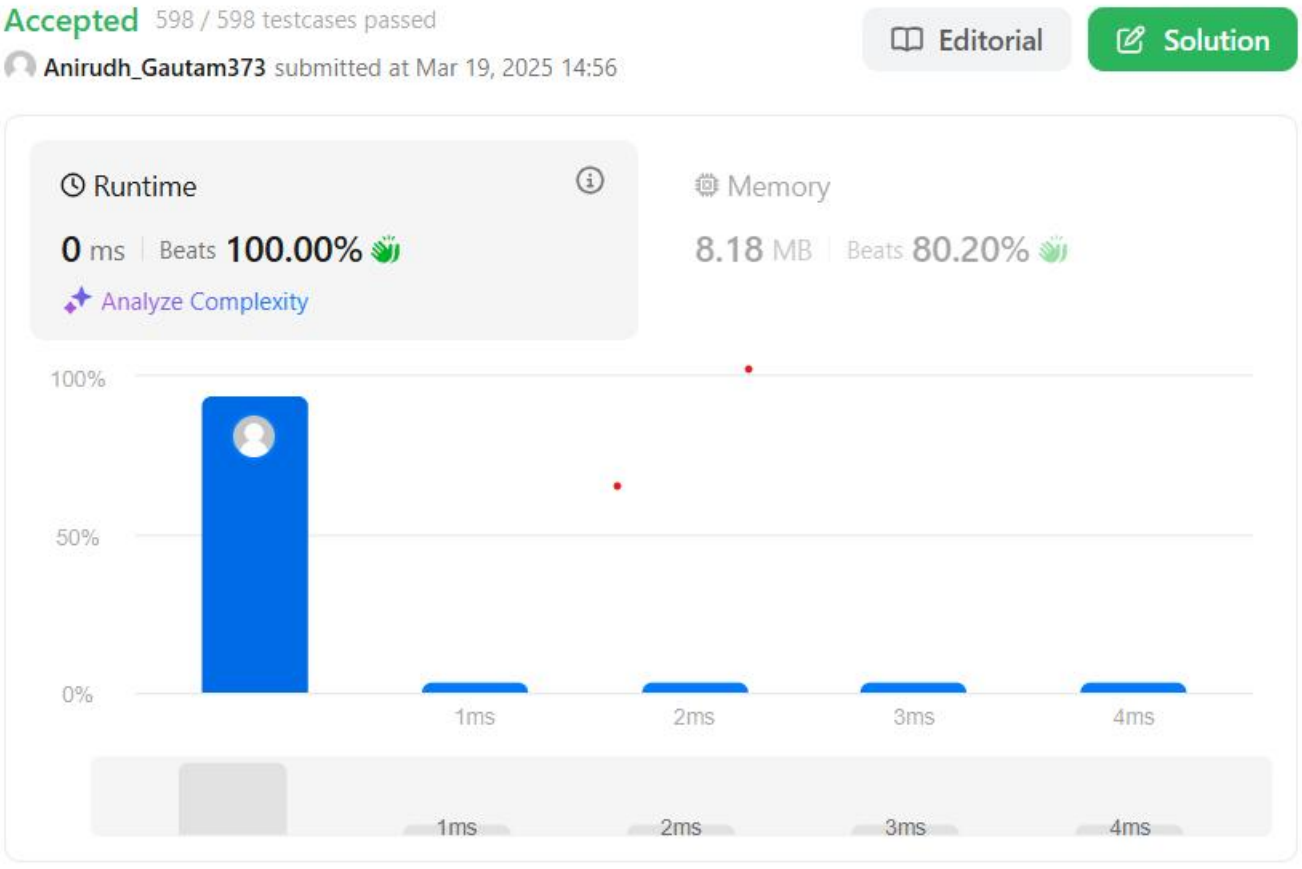
1. Convert Sorted Array to Binary Search Tree

```
class Solution {
public:
    TreeNode* sortedArrayToBST(vector<int>& nums) {
        return helper(nums, 0, nums.size() - 1);
    }

private:
    TreeNode* helper(vector<int>& nums, int left, int right) {
        if (left > right) return nullptr;
        int mid = left + (right - left) / 2;
        TreeNode* root = new TreeNode(nums[mid]);
        root->left = helper(nums, left, mid - 1);
        root->right = helper(nums, mid + 1, right);
        return root;
    }
};
```



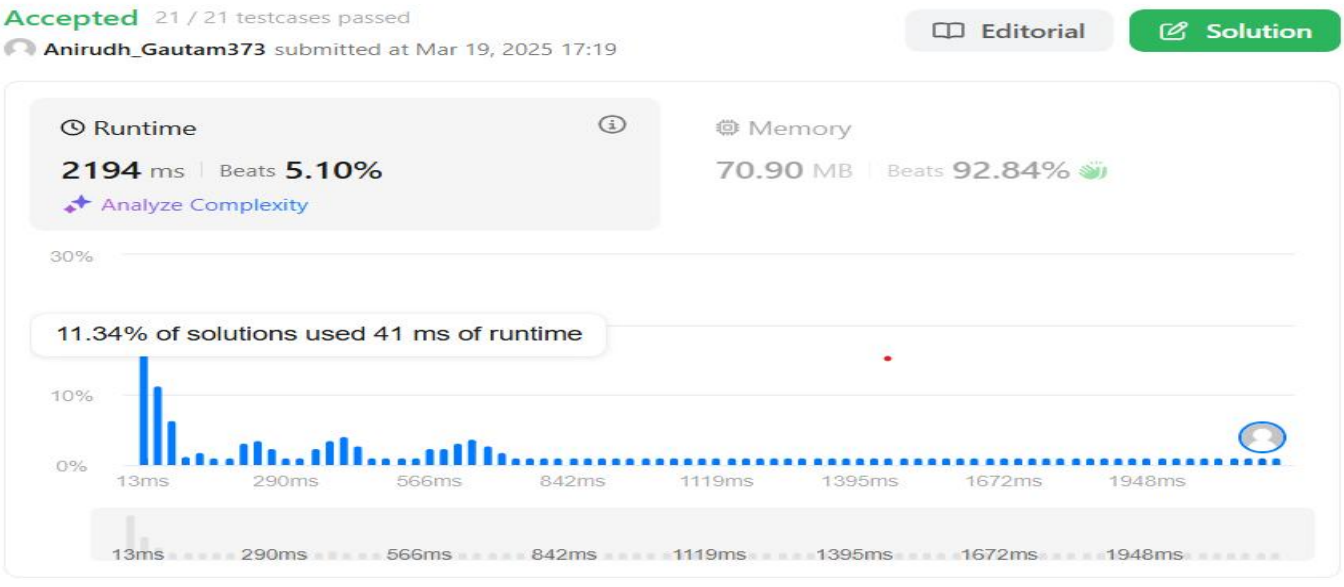
2. Number of Bits

```
class Solution {
public:
    int hammingWeight(uint32_t n) {
    int count = 0;

    while (n) {
        n &= (n - 1);
        count++;
    }

    return count;
}
};
```



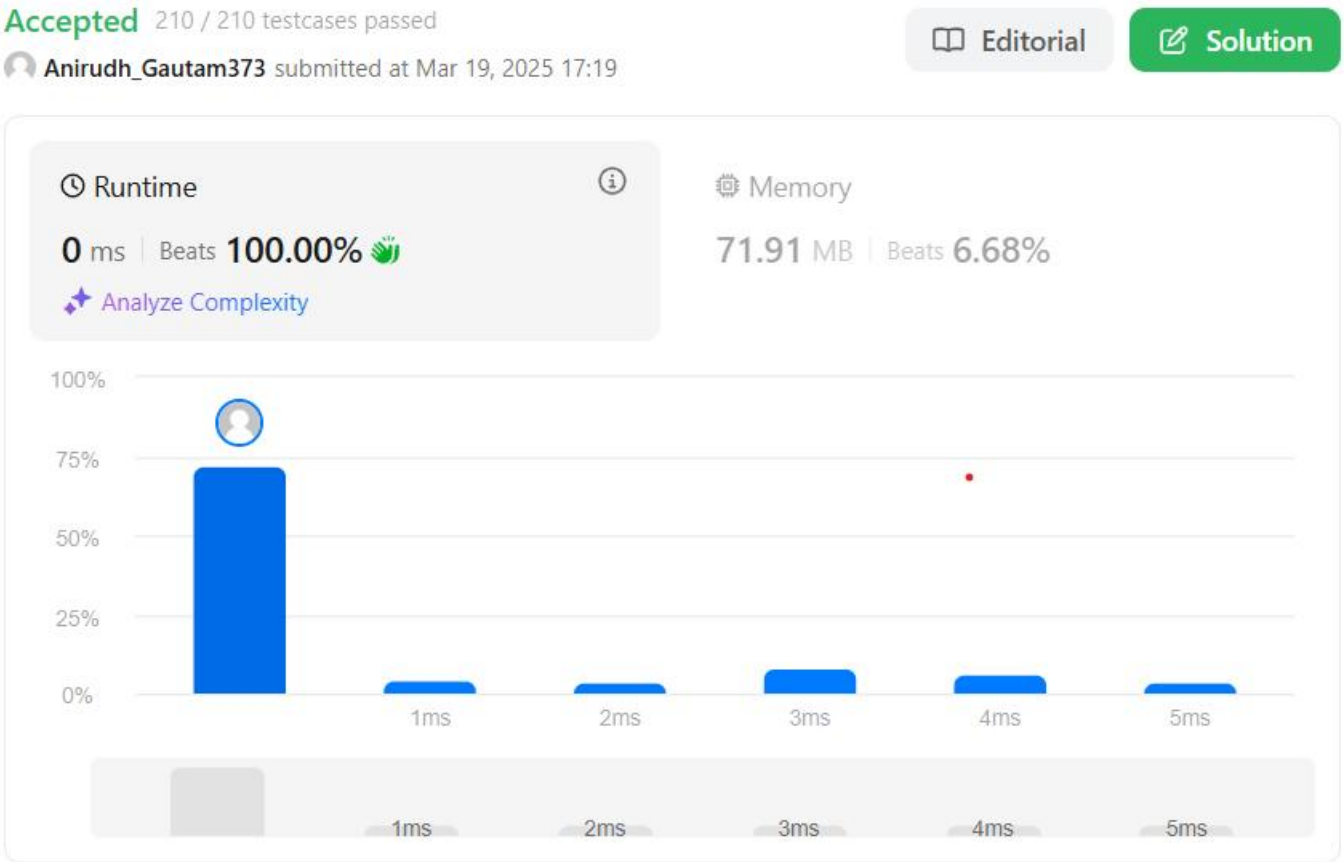
3. Sort an Array

```
class Solution {
public:
    int partition(vector <int>& arr,int l,int r){
    if (l >= r) return -1;
    int n=r-l;
    int pivot=l+rand()%n;
    swap(arr[l],arr[pivot]);
    int i=l+1;
    for(int j=l+1;j<r;j++){
        if(arr[j]<arr[l]){
            swap(arr[i],arr[j]);
            i++;
        }
    }
    swap(arr[l],arr[i-1]);
    return i-1;
  }
  void QuickSort(vector<int>& nums,int l,int r){
    if(l>=r){
        return;
    }
    int pivot=partition(nums,l,r);
    QuickSort(nums,l,pivot);
    QuickSort(nums,pivot+1,r);
  }
  vector<int> sortArray(vector<int>& nums) {
    int n=nums.size();
    QuickSort(nums,0,n);
    return nums;
  }};
```



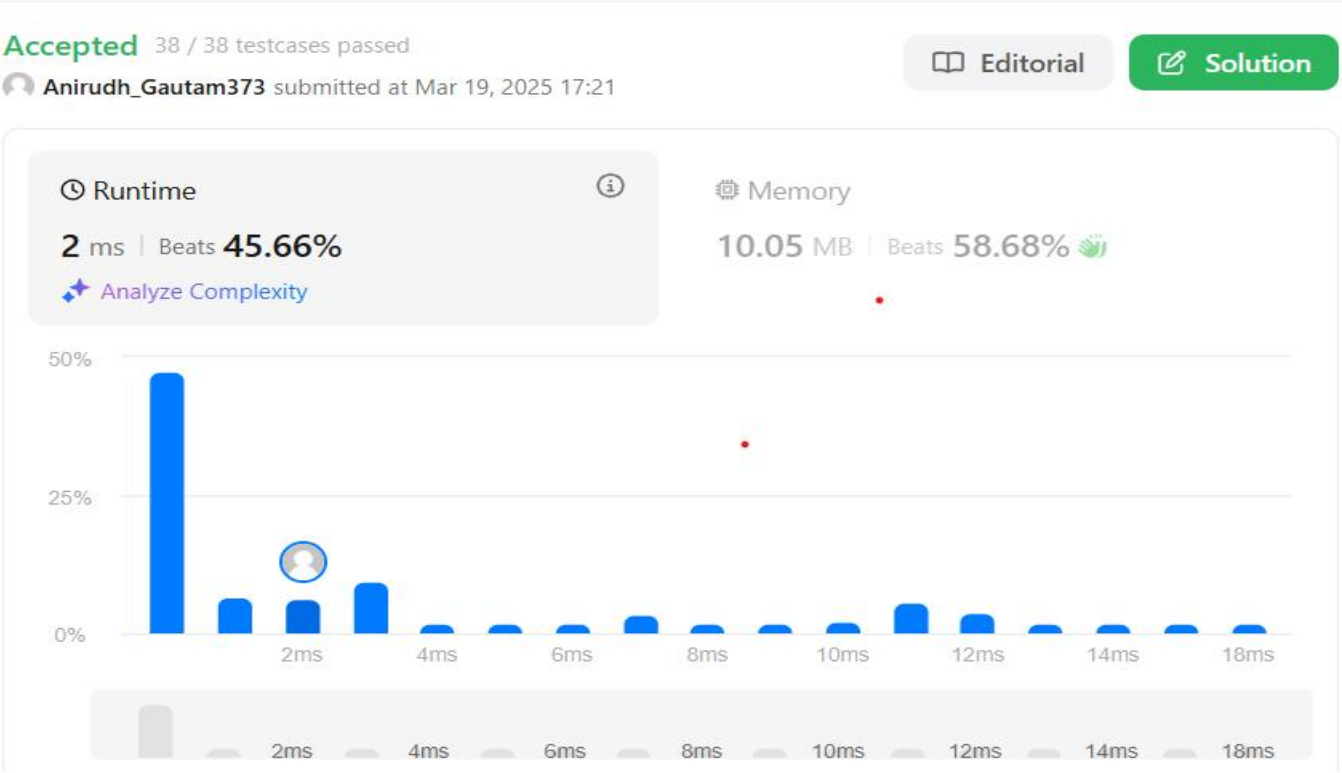
4. Maximum Subarray

```
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int res = nums[0];
        int total = 0;

        for (int n : nums) {
            if (total < 0) {
                total = 0;
            }

            total += n;
            res = max(res, total);
        }

        return res;
    }
};
```



5. Beautiful Array

```
class Solution {
public:
    vector<int> beautifulArray(int n) {
        vector<int> res{1};
        if(n==1) return res ;

        while(res.size()<n){
            vector<int> tmp;
            for(int i  :res){
                if(2*i <= n){
                    tmp.push_back(2*i);
                }
            }
             for(int i  :res){
                if(2*i -1 <= n){
                    tmp.push_back(2*i-1);
                }
            }
            res = tmp;
        }
        return res;
    }
};
```



6. Super pow

```
class Solution {
    const int base = 1337;
    int powmod(int a, int k)    {
        a %= base;
        int result = 1;
        for (int i = 0; i < k; ++i)
            result = (result * a) % base;
        return result;
    }
public:
    int superPow(int a, vector<int>& b) {
        if (b.empty()) return 1;
        int last_digit = b.back();
        b.pop_back();
        return powmod(superPow(a, b), 10) * powmod(a, last_digit) % base;
    }
};
```

7. The Skyline Problem

```
class Solution {
public:
    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
        vector<vector<int>> ans;
        multiset<int> pq{0};

        vector<pair<int, int>> points;

        for(auto b: buildings){
            points.push_back({b[0], -b[2]});
            points.push_back({b[1], b[2]});
        }

        sort(points.begin(), points.end());

        int ongoingHeight = 0;

        // points.first = x coordinate, points.second = height
        for(int i = 0; i < points.size(); i++){
            int currentPoint = points[i].first;
            int heightAtCurrentPoint = points[i].second;

            if(heightAtCurrentPoint < 0){
                pq.insert(-heightAtCurrentPoint);
            } else {
                pq.erase(pq.find(heightAtCurrentPoint));
            }

            // after inserting/removing heightAtI, if there's a change
            auto pqTop = *pq.rbegin();
            if(ongoingHeight != pqTop){
                ongoingHeight = pqTop;
                ans.push_back({currentPoint, ongoingHeight});
            }
        }

        return ans;
    }
};
```

Accepted 44 / 44 testcases passed

Anirudh_Gautam373 submitted at Mar 19, 2025 17:23

Editorial | Solution

Runtime

11 ms | Beats 86.82%

Analyze Complexity

Memory

28.69 MB | Beats 54.94%